CLAIMS

1. A computer system configured to:

A) provide a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, employ a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.

2. A computer system as defined in claim 1 wherein pushing occurs at one, bottom end of the queue, popping in accordance with the FIFO access mode occurs at the other, top end of the queue, and popping in accordance with the LIFO access mode occurs at the bottom end of the queue.

3. A computer system as defined in claim 1 wherein the queue accesses are circular.

4. A computer system as defined in claim 1 wherein the task-queue set includes of plurality of the task queues.

5. A computer system as defined in claim 4 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

6. A computer system as defined in claim 5 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

7. A computer system as defined in claim 6 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

8. A computer system as defined in claim 4 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

9. A computer system as defined in claim 8 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

10. A compiler/interpreter that, in response to signals representing instructions that define operations in which memory for data objects is allocated dynamically, generating

signals representing instructions that implement a garbage collector that operates in garbage-collection cycles of which each includes an operation that includes:

A) providing a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, employing a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.

11. A compiler/interpreter as defined in claim 10 wherein the task-queue set includes of plurality of the task queues.

12. A compiler/interpreter as defined in claim 11 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

13. A compiler/interpreter as defined in claim 10 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

14. A compiler/interpreter as defined in claim 13 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

15. For performing dynamically identified tasks, a method comprising employing a computer system to:

A) provide a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, employ a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.

16. A method as defined in claim 15 wherein pushing occurs at one, bottom end of the queue, popping in accordance with the FIFO access mode occurs at the other, top end of the queue, and popping in accordance with the LIFO access mode occurs at the bottom end of the queue.

17. A method as defined in claim 15 wherein the queue accesses are circular.

18. A method as defined in claim 15 wherein the task-queue set includes of plurality of the task queues.

19. A method as defined in claim 18 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

20. A method as defined in claim 19 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

21. A method as defined in claim 20 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

22. A method as defined in claim 18 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

23. A method as defined in claim 22 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

24. A storage medium containing instructions readable by a computer system to cause the computer system to:

A) provide a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, employ a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.

25. A storage medium as defined in claim 24 wherein pushing occurs at one, bottom end of the queue, popping in accordance with the FIFO access mode occurs at the other, top end of the queue, and popping in accordance with the LIFO access mode occurs at the bottom end of the queue.

26. A storage medium as defined in claim 24 wherein the queue accesses are circular.

27. A storage medium as defined in claim 24 wherein the task-queue set includes of plurality of the task queues.

28. A storage medium as defined in claim 27 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with

that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

29. A storage medium as defined in claim 28 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

30. A storage medium as defined in claim 29 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

31. A storage medium as defined in claim 27 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

32. A storage medium as defined in claim 31 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

33. A signal representing a sequence of instructions that, when they are executed by computer system, cause the computer system to:

A) provide a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, employ a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.

34. A signal as defined in claim 33 wherein pushing occurs at one, bottom end of the queue, popping in accordance with the FIFO access mode occurs at the other, top end of the queue, and popping in accordance with the LIFO access mode occurs at the bottom end of the queue.

35. A signal as defined in claim 33 wherein the queue accesses are circular.

36. A signal as defined in claim 33 wherein the task-queue set includes of plurality of the task queues.

37. A signal as defined in claim 36 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

38. A signal as defined in claim 37 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

39. A signal as defined in claim 38 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

40. A signal as defined in claim 36 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

41. A signal as defined in claim 40 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

42. A computer system comprising:

A) means for providing a task-queue set that includes at least one task queue in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each task queue, means for employing a separate execution thread associated therewith to:

i) select repeatedly between a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from the associated task queue in accordance with an access mode thus selected;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto the task queue task identifiers that identify any tasks thus found.